1 Cuckoo Hashing

Let us say we have an array $A$ of size $m = 4n$, two random hash functions $g, h$. We try to insert $x$ into $A[g(x)]$, potentially kicking out item already there and moving it. Note that this might cascade.

If a sequence of items moves goes on for $\geq C \cdot \lg n$ steps, we give up, pick new $g$ and $h$, and rebuild entire data structure.

Claim: $E$(time to insert $x) \leq O(1).

Proof: A cuckoo graph has $m$ vertices (one per cell of $A$) and $n$ edges (since for each $x$, we connect $g(x)$ to $h(x)$).

Consider the path we get from an insertion of $x$. We could get a simple path, a single cycle, or a double cycle. Let us define the following random variables: $T$, the runtime; $P_k$, the indicator random variable of a path being at least length $k$; $C_k$, the indicator random variable for single-cycle config of length $\geq k$; and $D$ the indicator for a random variable for having a 2-cycle config. Note also that the probability of the insertion process taking more than $N = C \log n$ steps implies that one of either $D$, $P_N$, or $C_N$ occurred. Therefore

We know that:

$$E[T] = E\left[\sum_k P_k + E\sum_k C_k + P(\text{go on for more than } C \log n \text{ steps}) \cdot n \cdot E[T]\right]$$

$$\leq E\left[\sum_k P_k + E\sum_k C_k + (P(D = 1) + E[P_N + E[C_N]) \cdot n \cdot E[T]\right]$$

(1)

Let us consider $E[P_k]$. Fix $x_2, x_3, \ldots, x_{k+1}$. Fix the assignment of the $(k+1)$ hash values to vertices. The probability we see exactly this path is $\frac{1}{m} \cdot \frac{1}{m^{k+1}} \cdot 2^k$. To do this, note that the number of total possible has values is $m^{k+1}$, the number of ways to choose edges is $n \cdot (n-1) \cdots (n-k+1) \leq n^k$.

Then, by union bound, we know that $E[P_k] \leq n^k \cdot m^{k+1} \cdot \frac{1}{m} \cdot \frac{1}{m^{k+1}} \cdot 2^k = \frac{1}{2^k}$.

Now, let us bound $C_k$. For $C_k$, let us define 3 types of edges (the forward edges, the backward edges, and edges on the subsequent path created by the other function). One of these must have $k/3$ edges, giving us a similar bound as the path analysis $E[C_k] \leq \frac{1}{2^{k/3}}$.

For $D$, we want $P(D = 1)$. Let $t$ denote the number of distinct vertices (which will also be the number of distinct edges, not including edges labeled with $x$) in the double cycle graph. Let $D_t$ be the indicator random variable for having a tour of this type with $t$ vertices. We know that

$$P(D = 1) = \sum_k P(D_t = 1)$$

(2)
Let us look for a particular configuration with \( t \) vertices. The probability we see this config is \( \frac{1}{m^2} \cdot \frac{1}{(m^2)^t} \cdot 2^t \) (the extra \( 1/m^2 \) comes from requiring \( x \) to hash to its two vertices). Union bounding over all configurations: we have at most \( m^t \) choices of vertices, at most \( n^t \) choices of edges, and at most \( t^3 \) choices for the start of the first cycle, the length of the first cycle, and the start of the second cycle. Thus

\[
P(D_t = 1) \leq t^3 \cdot \frac{(2mn)^t}{m^{2t+2}},
\]

which is at most \( (1/n^2)t^3 / 2^t \). Thus Eq. (2) converges and is \( O(1/n^2) \).

Now, in Eq. (1), the probability of going on for more than \( N \) steps is at most \( P(D = 1) + \mathbb{E} P_N + \mathbb{E} C_N \). By setting \( C \) large enough, this is \( O(1/n^2) \), dominated by the \( P(D = 1) \) term. Rearranging terms thus gives \( \mathbb{E} T = O(1) \), as desired.

### 2 Last Thing on Hashing

Let us talk about the “power of two choices.” Recall hashing w/ chaining. If we choose a perfect random hash function, with high probability, the length of the longest list is \( O\left(\frac{\lg n}{\lg \lg n}\right) \).

[azar brother karlin upfal sicomp ‘99] Pick 2 random hash functions \( g, h \). When inserting \( x \), place in the least loaded amongst \( A[g(x)] \) and \( A[h(x)] \). Now, with high probability, the heaviest bin has at most \( \frac{\ln \ln n}{\ln 2} + \Theta(1) \) items.

What about the power of \( d \) choices? We only improve by a constant factor, i.e., \( \frac{\ln \ln n}{\ln d} + \Theta(1) \) items in heaviest.

[Vöcking JACM ’03] Break up bins into \( d \) groups each of size \( n/d \). When insert item, check random locations in each group. Put in least loaded, break ties by placing in leftmost. Now, the maximum load is \( \Theta\left(\frac{\ln \ln n}{d}\right) \).

To see more, see survey by Mitzenmacher, Richa, Sitaraman.

**Intuition for power of 2 choices:**

Let \( B_i \) be the number of bins with load \( \geq i \). Let the height of \( x \), \( H(x) \) be such that \( x \) is he \( H(x) \)th item inserted into that bin.

Let \( Q_x \) be the indicator random variable for event that \( H(x) \geq i + 1 \). The probability that \( H(x) \geq i + 1 \) is at most \( \left( \frac{B_i}{n} \right)^2 \). So, if everything is as expected, \( B_{i+1} \leq n \cdot \left( \frac{B_i}{n} \right)^2 \), i.e., \( \left( \frac{B_{i+1}}{n} \right) \leq \left( \frac{B_i}{n} \right)^2 \).

Let’s say that \( \frac{B_{10}}{n} \leq \frac{1}{2} \). Then, \( \frac{B_{10+j}}{n} \leq \frac{1}{2^{2j}} \). We are done with \( B_{10+j}n < \frac{1}{n} \), which append when \( j \geq \lg \lg n \).

**More rigorous details:**

Below we outline how a more rigorous proof would go.

Define \( \alpha_6 = \frac{n}{2^5}, \alpha_{i+1} = \frac{e\sigma^2}{n} \). If \( E_i \) is the event that \( B_i \leq \alpha_i \), we will show that who all events \( E_i \) occur.

First, \( \mathbb{P}(E_6) = 1 \) because \( \frac{n}{2^5} > \frac{n}{6} \).
We would now like to show that $\mathbb{P}(\forall i E_i)$ is large. By the union bound, this is at least

$$1 - \sum_i \mathbb{P}(-E_i) \geq 1 - \mathbb{P}(-E_0) - \sum_i (\mathbb{P}(-E_{i+1}|E_i) + \mathbb{P}(-E_i))$$

$$1 - \sum_i (\mathbb{P}(-E_{i+1}|E_i) + \mathbb{P}(-E_i)) \quad (3)$$

It thus suffices to bound $\mathbb{P}(-E_{i+1}|E_i)$ and $\mathbb{P}(-E_i)$.

**Lemma 1.**

$$\mathbb{P}(-E_{i+1}|E_i) \leq \frac{\mathbb{P}(\text{Bin}(n, (\alpha_i/n)^2) > \alpha_{i+1})}{\mathbb{P}(E_i)}$$

where $\text{Bin}(n, p)$ is a binomial random variable with parameter $n, p$. That is, it is the sum of $n$ independent random Bernoulli random variables each with expectation $p$. Recall that a Bernoulli random variable is supported in $\{0, 1\}$.

**Proof.** For an item $j$, let the height $H(j)$ be such that $j$ is the $H(j)$th ball inserted into its bin. Let $Y_j$ be an indicator random variable for the event $H(j) \geq i + 1$. Then certainly $B_{i+1} \leq \sum_j Y_j$. It thus suffices to upper bound $\mathbb{P}(\sum_j Y_j > \alpha_{i+1}|E_i)$.

By Bayes’ rule,

$$\mathbb{P}(\sum_j Y_j > \alpha_{i+1}|E_i) = \frac{\mathbb{P}(\sum_j Y_j > \alpha_{i+1} \land E_i)}{\mathbb{P}(E_i)}$$

We then want to bound the numerator of the right hand side. Let $X_j$ be a Bernoulli random variable with $\mathbb{E}X_j = (\alpha_i/n)^2$. We will introduce the following “coupling” argument, which defines two sets of random variables $\{X_j\}, \{Y_j\}$ on the same probability space. Imagine picking uniform random variables $U_j, U'_j$ in $[0, 1)$. If both $U_j, U'_j \leq \alpha_i/n$, then we set $X_j$ to 1; else we set $X_j$ to 0. Now, imagine labeling the points $a_0 = 0/n, a_1 = 1/n, \ldots, a_n = n/n$ on the interval $[0, 1]$. As we will describe, these points correspond to the $n$ bins, in reverse sorted order by load. $U_j, U'_j$ when generated will land in $[a_{t-1}, a_t)$ and $[a_{t'-1}, a_{t'})$, respectively, for some $t, t'$. We then imagine placing a ball in the least loaded of bins $t, t'$ (recall $t = 1$ corresponds to the heaviest bin). If we are at a point where $E_i$ no longer holds, then we set $\hat{Y}_j = 0$. Otherwise we set $\hat{Y}_j = 1$ iff $H(j) \geq i + 1$ according to this process. Now observe two things:

(a) $\hat{Y}_j \leq X_j$ always (with probability 1). Therefore

$$\mathbb{P}(\sum_j \hat{Y}_j > \alpha_{i+1}) \leq \mathbb{P}(\sum_j X_j > \alpha_{i+1}) \quad (4)$$

(b) In any point in the above defined probability space where both $E_i$ and $\sum_j Y_j > \alpha_{i+1}$ hold, it also holds that $\sum_j \hat{Y}_j > \alpha_{i+1}$. Thus

$$\mathbb{P}(\sum_j Y_j > \alpha_{i+1} \land E_i) \leq \mathbb{P}(\sum_j \hat{Y}_j > \alpha_{i+1}) \quad (5)$$

Combining Eqs. (4) and (5) concludes the proof. □

3
We now $\Pr(E_0) = 1$ (and equivalently $\Pr(\neg E_0) = 0$). By an inductive argument, once we upper bound $\Pr(\neg E_i)$, we can invoke Lemma 1 to yield that upper bounding $\Pr(Bin(n, (\alpha_i/n)^2) > \alpha_{i+1})$ implies a bound on $\Pr(\neg E_{i+1}|E_i)$ (since in our inductive hypothesis we claim we have an upper bound on $\Pr(\neg E_i)$, and thus a lower bound on $\Pr(E_i) = 1 - \Pr(\neg E_i)$). One can bound $\Pr(Bin(n, (\alpha_i/n)^2) > \alpha_{i+1}) \leq e^{-C\alpha_i^2/n}$ via the Chernoff bound (calculation left as an exercise to the reader!). For the reader interested in seeing all the calculations worked out, see the notes at http://www.cs.berkeley.edu/~sinclair/cs271/n15.pdf.

## 3 Next Time

We will talk about data structures + amortized analysis, heaps (binomial and Fibonacci [2]), and splay trees [4].

For heaps, we store $n$ items w/keys (comparable). We can insert($x$), decreaseKey($x,k$), and deleteMin(). Dijkstra’s algorithm uses heaps in its implementation, and its runtime is $m \cdot$ insert + $m \cdot$ decreaseKey + $n \cdot$ deleteMin if there are $n$ vertices and $m$ edges. With binary heaps, all operations take $\log n$ time and thus Dijkstra runs in time $O((m+n) \log n)$. We will see that Fibonacci heaps support insert and decreaseKey each in $O(1)$ amortized time, and deleteMin in $O(\log n)$ amortized time, thus speeding up Dijkstra to $O(m+n \log n)$.

## References


